Documentation of my multiple attempts at this Bluetooth App:

As I listed in the final presentation document, I used FlutterFlow, a website that allowed me to easily create UI that is compatible with dart, a software language that can be used to build Android, IOS, and web applications, as well as machine code.

The UI was the easy part, after which I exported my code onto github, linked below:

- All of the files mentioned below will be found on this github to be refrenced

https://github.com/BibinSibi02/app

Following this began the setup, I had to install VS code to help me install flutter, the implementation that FlutterFlow built off of, and is a terminal-based tool that utilizes the dart programming language. From there, I had to install Android studio and its respective dependencies along with its proper extensions

- This follows the following setup documentation given by the official Flutter Website, with the setup taking multiple days:
- https://docs.flutter.dev/get-started/install/windows/mobile

This is where the real coding began, with the creation of

- lib/custom_code/actions/send_bluetooth_command.dart

This is a Custom Action in flutter terms, aka a method, which takes a string for a parameter. The UI contains two buttons, one which calls sendBluetoothCommand("unlock") to unlock over a Bluetooth connection, and sendBluetoothCommand("lock") to lock over a Bluetooth connection. The script on the Raspberry Pi has initially explained to me that it received any data sent over bluetooth, and then filters the data for the "unlock" and "lock" keywords that my Custom Action was designed to send on UI press.

The app also did not need to initially create and verify the bluetooth connection, and instead utilized the existing Bluetooth to the Rpi, allowing for the code to be clean, without any UI or code dedicated to searching for and establishing the connection, but only for validating and utilizing an existing connection.

The concept seemed simple enough right? This is where I introduce the next key elements of the app for me to create and edit

- pubspec.yaml, the package declaration and app declaration file
- android/app/src/debug/AndroidManifest.xml, the permissions list for any android application
- ios/ImageNotification/Info.plist, the permissions list for any IOS application
- **ALL** of the .gradle files found in the android folder

To start off with the easiest part, the Info.plist file is needed to access the bluetooth connections and functions of any IOS device, which needed the following permissions:

```
<!-- Bluetooth Permissions for iOS -->
<key>NSBluetoothAlwaysUsageDescription</key>
<string>We need Bluetooth to communicate with the lock.</string> <!-- Required for iOS 13+ -->
<key>NSBluetoothPeripheralUsageDescription</key>
<string>We need Bluetooth to communicate with the lock.</string> <!-- Required for iOS 12 and below -->

<!-- Location Permissions for Bluetooth (needed for iOS 13 and above) -->
<key>NSLocationWhenInUseUsageDescription</key>
<string>We need location access to detect nearby Bluetooth devices.</string> <!-- Required for Bluetooth scanning -->

<key>NSLocationAlwaysUsageDescription</key>
<string>We need location access to detect nearby Bluetooth devices even when the app is in the background.</string> <!-- Needed for background Bluetooth scanning -->

<!-- Background modes for Bluetooth scanning and peripherals -->
<key>UIBackgroundModes</key>
<array>
  <string>bluetooth-central</string>  <!-- For scanning and connecting to Bluetooth devices -->
  <string>bluetooth-peripheral</string> <!-- If your app also communicates as a Bluetooth peripheral -->
</array>
```

The Info.plist file came with a couple other default permissions, that can be referenced on the github.

Unfortunately, the Info.plist file did not get updated much, as developing an IOS app was ruled out too late due to the unfortunate circumstance that I needed to pay $99 for a developer account to even publish the app in test_flight.

The next part, in order to directly access the bluetooth connection and functions of an android device, the AndroidManifest.xml needed the following permissions:

```
<!-- Bluetooth Permissions -->
<!-- Permission to use Bluetooth -->
<uses-permission android:name="android.permission.BLUETOOTH" />

<!-- Permission to manage Bluetooth devices (discover and pair) -->
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

<!-- Permission to access location (needed for Bluetooth scanning) -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

<!-- Permission for background location access (needed for Bluetooth scanning in the background) -->
<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />

<!-- Permission for scanning Bluetooth Low Energy (BLE) devices -->
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />

<!-- Permission for connecting to Bluetooth devices -->
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
```

A couple other permissions were added later, which can be referenced on github. This file will be important later on, as implementing an APK for an android app became the only other option left.

Finally, the most important part, the pubspec.yaml file, which not only listed the version and the environment, but also the dependencies, a small snippet of which is pasted below.

```
# https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html
version: 1.0.0+1

environment:
  sdk: ">=3.0.0 <4.0.0"

dependencies:
  flutter:
    sdk: flutter
  flutter_localizations:
    sdk: flutter
  flutter_web_plugins:
    sdk: flutter
  auto_size_text: 3.0.0
  cached_network_image: 3.4.1
  cached_network_image_platform_interface: 4.1.1
  cached_network_image_web: 1.3.1
  collection: 1.18.0
  firebase_core: 3.8.0
  firebase_core_platform_interface: 5.3.0
  firebase_core_web: 2.18.1
  firebase_performance: 0.10.0+10
  firebase_performance_platform_interface: 0.1.4+46
  firebase_performance_web: 0.1.7+4
  flutter_animate: 4.5.0
  flutter_cache_manager: 3.4.1
  font_awesome_flutter: 10.7.0
  from_css_color: 2.0.0
  go_router: 12.1.3
  google_fonts: 6.1.0
  intl: 0.19.0
  json_path: 0.7.2
  page_transition: 2.1.0
  path_provider: 2.1.4
  path_provider_android: 2.2.10
  path_provider_foundation: 2.4.0
  path_provider_linux: 2.2.1
```

The full file can be referenced on github.

Before I could even build the APK for the app to test, this pubspec.yaml file is where I spent countless hours of trial and error testing.

To build the app using flutter, I first had the app files on my desktop, which I had ssh-ed into via Powershell. Since I had spent days tying to install flutter in my system, I had access to flutter's build tools, including the following:

- flutter doctor - a command that essentially showed me the versions of gradle, java, jdk, and other dependencies allowing for easier diagnosis of compatibility errors in pubspec.yaml to an extent

- flutter pub get - a command which essentially fetched all the dependencies found in the pubspec.yaml file
- flutter clean - a command which cleaned out any compilations and dependencies fetched
- flutter build apk - - release - a command that attempted to build the APK

To understand the root of my problems, we need to visit the last part of an android APK, the .gradle files, all of which can be found on github.
Unfortunately, gradle integration and package compatibility issues was one that I had to manually fix, piece by piece through trial and error, because by downgrading or upgrading a version of something in the pubspec.yaml file to fix an error with compatibility issues with gradle, this would then cause more errors with package compatibility within the packages found in the pubspec.yaml. This essentially meant that any changes made always resulted in a ripple effect, affecting other packages around it. Ultimately, I was unable to resolve this issue to make the bluetooth packages flutter_blue and the flutter_blue_plus, both of which I tried separately.